Type Qualifiers in Hive

Intro

Hive will need to support some kind of type qualifiers/parameters in its type metadata to be able to enforce type features such as decimal precision/scale or char/varchar length and collation. This involves changes to the PrimitiveTypeEntry/TypeInfo/ObjectInspectors, possibly metastore changes,
My impression is that the actual enforcement of the type qualifiers should be done by the ObjectInspectors/Converters/casts operations. It should be ok to do col * col when col is a decimal(2) value of 99, it would fail if you try to cast the result to decimal(2) or try to insert it to a decimal(2) column.

Initial prototype work

There is some initial work on this in an initial patch for HIVE-4844. There is a BaseTypeParams object to represent type parameters, with VarcharTypeParams as a varchar-specific subclass containing the string length. The PrimitiveTypeEntryTypeInfo/ObjectInspectors are augmented to contain this BaseTypeParams object if the column/expression has type parameters. There also needed to be additional PrimitiveTypeEntryTypeInfo /ObjectInspectors factory methods which take a BaseTypeParams parameter.

Some issues/questions from this:

- There are some cases where Hive is trying to create a PrimitiveTypeEntry or ObjectInspector based on a Java class type. Such as ObjectInspectorFactory.getReflectionObjectInspector(). In these cases, there would be no data type params available to add to the PrimitiveTypeEntry/ObjectInspector, in which case we might have to default to some kind of type attributes - max precision decimal or max length char/varchar. This happens in a few places:
    - TypedSerDe (used by ThriftByteStreamTypedSerDe). Might be ok since if it's just using Thrift types.
    - S3LogDeserializer (in contrib). Might be ok, looks like it is only a deserializer, and for a custom S3 struct.
    - MetadataTypedColumnsetSerDe. Might be ok, looks it might just use strings.
    - GenericUDFUtils.ConversionHelper.ConversionHelper(), as well as GenericUDFBridge. This is used by old-style UDFs, in particular for the return type of the UDF. So in the general case it is not always possible to have type parameters for the return type of UDFs. GenericUDFs would be required if we want to be able to return a char length/decimal precision as part of the return type metadata, since they can customize the return type ObjectInspector.
- If cast operators remain implemented as UDFs, then the UDF should probably be implemented as a Generic UDF so that the return type ObjectInspector can be set with the type params. In addition, the type parameters need to be somehow passed into the cast UDF before its initialize() method is called.
- Hive code does a lot of pointer-based equality using PrimitiveTypeEntry/TypeInfo/ObjectInspector objects. So a varchar(15) object inspector is not equal to a varchar(10). This may have some advantages such as requiring conversion/length enforcement in this case, but it seems like this may not always be desirable behavior.

MetaStore Changes

There are a few different options here:

No metastore changes

The type qualifiers could simply be stored as part of the type string for a column. The qualifiers would be validated during when creating/altering the column, and they would need to be parsed when creating TypeInfo/ObjectInspectors. This approach has the advantage that no additional metastore changes would be needed, though it would be more difficult to query these type attributes if someone is querying the metastore directly, since parsing of the type string is required.

Add additional columns to COLUMNS_V2 table in metastore

This approach would be similar to the attributes in the INFORMATION_SCHEMA.COLUMNS that some DBMS catalog tables have, such as those listed below:

<pre>

CHARACTER_MAXIMUM_LENGTH	bigint(21) unsigned	YES		NULL	
CHARACTER_OCTET_LENGTH	bigint(21) unsigned	YES		NULL	
NUMERIC_PRECISION	bigint(21) unsigned	YES		NULL	
NUMERIC_SCALE	bigint(21) unsigned	YES		NULL	
CHARACTER_SET_NAME	varchar(32)	YES		NULL	
COLLATION_NAME	varchar(32)	YES		NULL	

</pre>

We could add new columns to the COLUMNS_V2 table for any type qualifiers we are trying to support (initially looks like CHARACTER_MAXIMUM_LENGTH, NUMERIC_PRECISION, NUMERIC_SCALE). Advantages to this would be that it is easier to query these parameters than the first approach, though types with no parameters would still have these columns (set to null).

New table with type qualifiers in megastore

Rather than having to change the COLUMNS_V2 table we could have a new table to hold the type qualifier information. This would mean no additions to the existing COLUMNS_V2 table, and non-parameterized types would have no rows in this new table. But it would mean an extra query to this new table any time we are fetching column metadata from the metastore.